Generalizing to New Cup Positions in the Game of Beer Pong Using Contextual Probabilistic Movement Primitives

Alymbek Sadybakasov
Autonomous Systems
TU Darmstadt
`alymbek.sadybakasov@stud.tu-darmstadt.de`

Boris Belousov
IAS
TU Darmstadt
`boris@robot-learning.de`

Abstract

Contextual Probabilistic Movement Primitives (Contextual ProMPs) extend ProMPs by adding context variables — i.e., variables that do not change during trajectory execution — to the state representation of a system. In this paper, we use Contextual ProMPs in the beer pong task to generalize demonstrated throwing movements to new locations of the cup. Furthermore, we compare different encodings of the context variables, i.e., position of the cup. We approximate the context by a basis expansion in the weight-space of ProMPs using classical conditioning of Contextual ProMPs. In addition, we put the context directly into the state vector using Contextual Linear Regression (CLR), which is equivalent to conditioning of Contextual ProMPs. Using an approximated context together with conditioning of Contextual ProMPs, we achieved a success rate of 70% of hits and 20% of nearly hits. In contrast, CLR was not that convenient in generating such successful throwing movements, achieving only 2 successful and 3 nearly successful hits out of overall 10 attempts using the same set of demonstrations.

1 Introduction

Probabilistic Movement Primitives (ProMPs, Paraschos et al. [2013]) is a class of models that can be viewed as a parametric probability distribution over the space of trajectories. This representation allows performing various probabilistic operations to adjust a movement primitive to new circumstances. For instance, a via-point can be specified through conditioning, and coupling between joints can be encoded via a distribution over their trajectories. Moreover, movement trajectories can be conditioned on external variables called context using Contextual ProMPs, e.g., the location of a cup in beer pong.
We use beer pong[1] as the task on which we apply contextual ProMPs. A typical setup of a beer pong game consists of a ball and a pre-defined amount of the cups filled with beer for each team. Each team aims then to land the ball in a cup of the other team. A cup with a caught ball is counted as eliminated. As a first attempt, we simplify the rules by leaving only one cup with no pre-filled beer. In future work, the robot should be able to play a full game.
Each team can use different techniques in order to land the ball in the cup: an arc shot, a fastball, and a bounce shot. As a first attempt at solving this problem, we only use the arc shots to make the throwing movements. In a future work, this approach can be extended to contextual policies that will choose among different types of shots based on the current game situation.
In this paper, we apply the framework of contextual ProMPs to the game of beer pong in order to generalize a throwing movement to new locations of the cup. To make contextual ProMPs work in

[1]A more detailed explanation of the rules can be found at `https://en.wikipedia.org/wiki/Beer_pong`

the game of beer pong, we require very precise demonstrations. We describe the way to obtain the demonstrations, which turns out to produce very precise demonstrations. With this set of demonstrations, we achieve promising results with contextual ProMPs, indicating that contextual ProMPs are very convenient to solve the problems that contain context variables.
In addition, we compare the generalization quality of Contextual Linear Regression (CLR) in the weight-space of ProMPs with contextual ProMPs.
Finally, we compare CLR in the weight-space of ProMPs with CLR in the weight-space of Dynamic Movement Primitives (DMPs, Schaal [2003]).

2 Related Work

Several attempts have been made to learn beer pong in a simulation. Wieland and Hoppe [2013] optimized a throwing movement parameterized by a DMP using the POWER algorithm. Wagner and Schmitt [2013] extend their work by enabling generalization to new cup locations through contextual POWER. Due to the complexity of the problem, only limited generalization ability was achieved. We propose to learn a single-throw movement by imitation learning and generalize to new cup locations by contextual ProMP conditioning. Compared to the previous work, this approach can be used entirely on a physical system, without requiring precise simulation of the ball and robot dynamics.
There are several more approaches that can be applied to the beer pong problem. For instance, Maeda et al. [2014] used ProMPs to model human–robot interaction by conditioning on the human movement. We are closely following this work by replacing the human movement with the position of the cup. Contextual Policy Search (Kupcsik et al. [2017]) can be applied to improve generalization in case if a good simulator of the environment is available.

3 Probabilistic Movement Primitives

We provide a short introduction into Probabilistic Movement Primitives (ProMPs) and describe the way how ProMPs can be conditioned on a subset/context of a state. In addition, we discuss the relationship between contextual ProMP and CLR within the weight-space of ProMPs.

3.1 Basics

In ProMPs, a slice of a trajectory at time t is assumed to be linear in features. To make the parameters learnable, a Gaussian noise is added to the model, i.e.,

$$y_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \boldsymbol{\Phi_t^T w} + \boldsymbol{\epsilon_y},$$

where $\boldsymbol{\Phi_t} = [\phi, \dot{\phi}]$ defines a diagonal and time-dependent basis function matrix and $\boldsymbol{\epsilon_y} \sim \mathcal{N}(0, \boldsymbol{\Sigma_y})$ is zero-mean i.i.d. Gaussian noise.$\phi_i = \exp\{-\frac{(z-c_i)^2}{2h}\}$ denotes a Gaussian radial basis function. The probability of observing the trajectory τ is then given by $p(\tau|\boldsymbol{w}) = \prod_t \mathcal{N}(\boldsymbol{y_t}|\boldsymbol{\Phi_t^T w}, \boldsymbol{\Sigma_y})$.
Temporal modulation and scaling are achieved by introducing the phase variable z which must be a monotonically increasing function. The basis functions then directly depend on z instead of time.
Encoding the coupling between n joints is achieved by stacking the weights of each joint into one vector $\boldsymbol{w} = [\boldsymbol{w_t^T}, \ldots, \boldsymbol{w_n^T}]^T$ and building a block-diagonal matrix $\boldsymbol{\Psi}$ with $\boldsymbol{\Phi_t}$ as its diagonal entries for each dimension.
Assuming a Gaussian distribution over the weights $\boldsymbol{w}$, the probability of observation y at time t is then given by

$$p(\boldsymbol{y_t}|\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{y_t}|\boldsymbol{\Psi_t w}, \boldsymbol{\Sigma_y}) = \mathcal{N}(\boldsymbol{y_t}|\boldsymbol{\mu_w}, \boldsymbol{\Sigma_y}).$$

The learning parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}\}$ can be acquired by maximum likelihood estimation.

3.2 Conditioning

Assuming a Gaussian distribution over trajectories, ProMPs can be easily conditioned on some desired state $\boldsymbol{y_t^*}$, e.g., new desired position of the cup. For conditioning, a desired observation $\boldsymbol{x_t} = [\boldsymbol{y_T^*}, \boldsymbol{\Sigma_y^*}]$

is added, and, after applying the Bayesian theorem, we have $p(\boldsymbol{w}|\boldsymbol{x}_t^*) = \mathcal{N}(\boldsymbol{y}_t^*|\boldsymbol{\Psi}^T\mathbf{w}, \boldsymbol{\Sigma}_y^*)\boldsymbol{p}(\boldsymbol{w})$. To find $p(\boldsymbol{w}|\boldsymbol{x}_t^*)$, we update the mean and the variance:

$$\begin{aligned}\boldsymbol{\mu}_w^+ &= \boldsymbol{\mu}_w^- + \boldsymbol{\Sigma}_w\boldsymbol{\Psi}_t(\boldsymbol{\Sigma}_y^* + \boldsymbol{\Psi}_t^T\boldsymbol{\Sigma}_w\boldsymbol{\Psi}_t)^{-1}(\boldsymbol{y}_t^* - \boldsymbol{\Psi}_t^T\boldsymbol{\mu}_w),\\ \boldsymbol{\Sigma}_w^+ &= \boldsymbol{\Sigma}_w^-\boldsymbol{\Psi}_t(\boldsymbol{\Sigma}_y^* + \boldsymbol{\Psi}_t^T\boldsymbol{\Sigma}_w^-\boldsymbol{\Psi}_t)^{-1}\boldsymbol{\Psi}_t^T\boldsymbol{\Sigma}_w^-.,\end{aligned} \tag{1}$$

where $^-$ and $^+$ stand for previous and new calculated variables respectively. The important property of this conditioning is that the generated trajectory stays within the original distribution if the new desired goal position lies in the distribution as well. We can also condition on the via-points at the desired time steps, such that the generated trajectory will be including those via-points. For that, we update the mean and the variance for each desired via-point.

3.3 Contextual ProMPs

As stated before, we can condition partially on a context $\boldsymbol{c}$ (Paraschos et al. [2017], Maeda et al. [2014]). This conditioning is done by constructing the basis function matrix $\boldsymbol{\Psi}_t$ from Eq. (1) to contain only the variables that participate in the conditioning, and replacing $\boldsymbol{y}_t^*$ with $\boldsymbol{c}$. In case of the beer pong problem, we let $\boldsymbol{\Psi}_t$ contain the variables only in the last two diagonal elements and set the first 7 diagonal elements to zero. We use only x- and y-coordinates to represent the cup positions and let $\boldsymbol{c}$ contain only the context variables as well, setting other variables to 0. We ignore the z-coordinate since the cup stays on the same table and, thus, does not change its z-coordinate. Pluging both $\boldsymbol{\Psi}_t$ and $\boldsymbol{y}_t^*$ into Eq. (1), we obtain a conditioned distribution. This approach can be viewed as *Contextual ProMPs*, with

$$\begin{aligned}\boldsymbol{\mu}_{w|c} &= \boldsymbol{\mu}_w + \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}(\boldsymbol{c} - \boldsymbol{\mu}_c),\\ \boldsymbol{\Sigma}_{w|c} &= \boldsymbol{\Sigma}_{ww} - \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}\boldsymbol{\Sigma}_{cw},\end{aligned} \tag{2}$$

from where we obtain the weights $\boldsymbol{w} = \boldsymbol{\mathcal{N}}(\boldsymbol{\mu}_{w|c}, \boldsymbol{\Sigma}_{w|c})$.

3.4 Relationship Between the Mean of Contextual ProMPs and CLR

There is a relationship between the mean of ProMP and linear regression, which has been also noted in [Paraschos et al., 2017, Section 4.3.2]. We expand this observation by deriving the mean of contextual ProMP in a different way and comparing it to minimizing the objective function of CLR. Using [Bishop, 2006, Eq. 2.96], we rewrite the mean $\boldsymbol{\mu}_{w|c}$ from Eq. (2) as

$$\begin{aligned}\boldsymbol{\mu}_{w|c} &= \boldsymbol{\mu}_w + \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}(\boldsymbol{c} - \boldsymbol{\mu}_c),\\ &= \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}\boldsymbol{c} + (\boldsymbol{\mu}_w - \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}\boldsymbol{\mu}_c),\\ &= \boldsymbol{A}\boldsymbol{c} + \boldsymbol{b},\end{aligned}$$

from where we have

$$\boldsymbol{A} = \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}. \tag{3}$$

We observe that the conditional mean is a linear function of the context, where $\boldsymbol{\Sigma}_{wc}$ and $\boldsymbol{\Sigma}_{cc}$ can be estimated as

$$\begin{aligned}\hat{\boldsymbol{\Sigma}}_{wc} &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{w}_i\boldsymbol{c}_i^T,\\ \hat{\boldsymbol{\Sigma}}_{cc} &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{c}_i\boldsymbol{c}_i^T.\end{aligned} \tag{4}$$

The estimate of the vector $\boldsymbol{b}$ can be found in the same way.
To apply CLR, we obtain one trajectory per cup position c and fit the basis functions of ProMPs to this trajectory in order to obtain the weights $\boldsymbol{w}$. After collecting several weight vectors that correspond to different cup positions, we assume that the vector is a linear function of the cup position, i.e., $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{c} + \boldsymbol{b}$. We ignore the additive constant b for simplicity and, thus, have

$$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{c}.$$

Putting all column vectors $\boldsymbol{w_i}$ into a matrix W and putting all row vectors $\boldsymbol{c_i^T}$ in a matrix C, the linear regression objective function can be stated as

$$J = \mathrm{tr}\{(\boldsymbol{W} - \boldsymbol{AC})^{\boldsymbol{T}}(\boldsymbol{W} - \boldsymbol{AC})\}. \tag{5}$$

Minimizing 5 with respect to A, we obtain

$$\boldsymbol{A} = \boldsymbol{WC^T}(\boldsymbol{CC^T})^{-\boldsymbol{1}}. \tag{6}$$

We rewrite matrix products from Eq. (6) as

$$\boldsymbol{WC^T} = \sum_{i=1}^{N} \boldsymbol{w_i c_i^T},$$

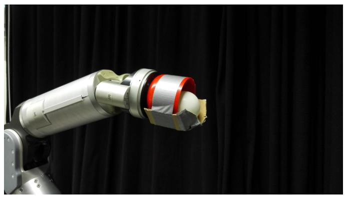
$$\boldsymbol{CC^T} = \sum_{i=1}^{N} \boldsymbol{w_i c_i^T},$$

and, after substituting them back into Eq. (6), obtain a similar equation as after substituting Eq. (4) into Eq (3). Note that in CLR, we do not normalize the product parts of the matrix $\boldsymbol{A}$. We will study the influence of the normalization factor in the experiments on the real robot in the next Section.

4 Experiments

We created a custom end effector that prevents the ball from being falling straight before the throwing movement. However, it turned out to be hard to model how exactly the trajectory of the ball depends on the trajectory of the robot. Nevertheless, experiments on the hardware showed a great repeatability of successful throwing movements. Therefore, we decided to evaluate both contextual ProMPs and CLR on a real robotic system without bothering with simulation.

4.1 Environment


(a) End effector with a ball inside it. The end effector consists of a cup to which a simple holder has been attached.

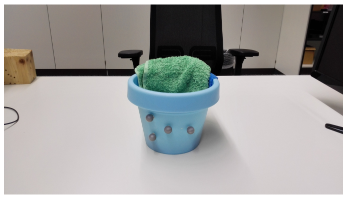

(b) Cup with markers that are used for tracking its position. At least 4 markers are needed to create a rigid body in OptiTrack.

Figure 1: Experimental setup.

We did our experiments on the Barett WAM[2] - a highly dexterous robotic arm with 7 degrees of freedom. An SL simulator has been used to record and evaluate the computed trajectories before executing it on the robot. To enable fast prototyping, a Robcom interface has been used which allows sending the trajectories from Python to the robot via SCTP.
In order to be able to throw the ball, a simple end effector has been built on the last joint. The end effector consists of a cup and a holder (Figure 1a). The holder prevents the ball from being falling in case the cup is tilted for more than 90 degrees. A tracking system Optitrack has been used for recording the cup positions (Figure 1b), which was treated as a rigid body. We tracked the ball in the initial experiments, but decided against doing this, since tracking the cup position was already sufficient enough for our experiments. The joint positions were recorded using SL with the help of Robcom. New trajectories were computed using Python and sent to the robot via the Robcom interface.

[2] http://barrett.com/products-arm.htm

We evaluated two types of throwing movements - from the bottom and from the top. Throwing the ball from the bottom required too high accelerations which could damage the robot. We, thus, decided to use only the throws from the top with accelerations that could be reproduced by the robot without damaging it.

4.2 Obtaining Demonstrations from the Real Robot

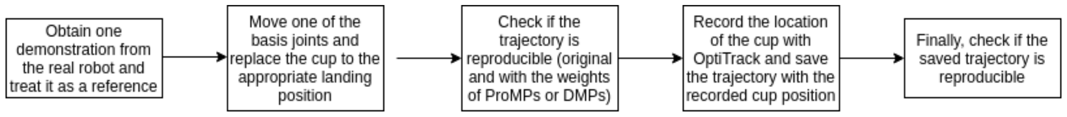
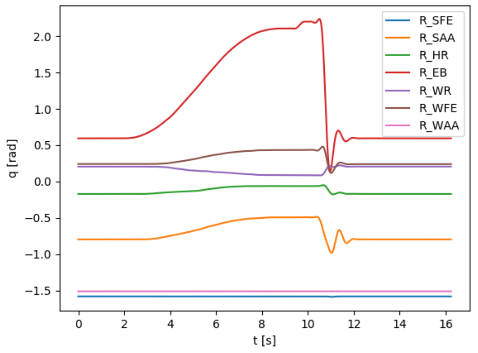
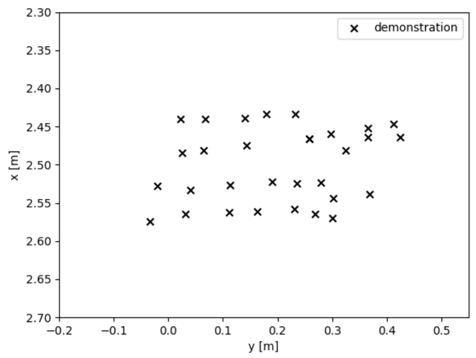

Obtain one demonstration from the real robot and treat it as a reference
Move one of the basis joints and replace the cup to the appropriate landing position
Check if the trajectory is reproducible (original and with the weights of ProMPs or DMPs)
Record the location of the cup with OptiTrack and save the trajectory with the recorded cup position
Finally, check if the saved trajectory is reproducible

Figure 2: An illustration of the workflow that we use to obtain demonstrations.



(a) A trajectory that produces a successful throwing movement. This trajectory serves as a template for all demonstrations.



(b) Locations of the cup on the table for which we provide demonstrations.

Figure 3: An illustration of the demonstrations that has been obtained on the real robot.

Obtaining demonstrations on the real robot is a crucial part of our learning process, as we need to pay attention to a proper correspondence between joint trajectories and locations of the cup. We propose a straightforward workflow to obtain demonstrations from a single demonstration provided by a human on the real robot.
We use kinesthetic teaching to record throwing movements and the OptiTrack system to track positions of the cup. Instead of recording trajectories for each cup position separately, we record only one successful throwing movement and filter the resulting joint trajectories using Butterworth filter (Butterworth [1930]). We generate the needed amount of demonstrations using this single demonstration by moving the base joints *SFE* or *SAA*[3] in the direction of new locations of the cup. In-between, we check if the resulting trajectories (plain and computed by the weights of ProMPs/DMPs) can be reproduced by the robot. We need 300 basis functions for ProMPs and 100 basis functions for DMPs in order to reproduce the throwing movement. The high amount of the basis functions becomes clear if we take a look at the joint trajectories of a throwing movement (Figure 3a). We can observe there a high peak in the trajectory of elbow-joint (R_EB) which is needed to produce the throwing movement. This high peak can only be reproduced by a big number of basis functions.
Once a plain trajectory and a trajectory computed by the weights of ProMPs/DMPs are reproducible at least twice, we record the position of the cup using OptiTrack. We omit the z-position of the cup since it always stays on the same table and, thus, does not move in vertical directions. Locations of the cup for which we provide demonstrations are illustrated in Figure 3b.
Figure 2 illustrates the complete workflow. A clear advantage of this approach is that it requires less time to record multiple demonstrations as we only need one template demonstration provided

[3]The acronyms SFE and SAA stand for Shoulder Flexion-Extension and Shoulder Abduction-Adduction. SFE is responsible for rotating the whole robot around its base, while SAA rotates the arm itself.

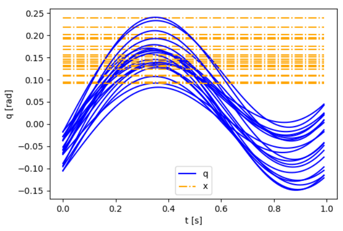
by a human on the real robot, while the rest of demonstrations are obtained using this template demonstration. Another advantage is that we immediately ensure that the trajectories and the corresponding cup positions are well synchronized. Furthermore, an exact reproduction of the demonstration by the radial basis functions is no more needed, since we can always place the cup to that position where the ball lands after executing the demonstration, and, later, record this position.

4.3 Contextual ProMP

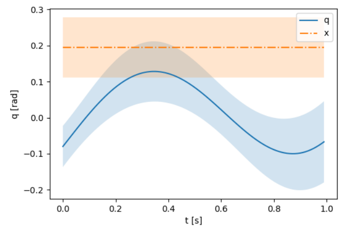
We implemented contextual ProMP in the way such that it could be conditioned on some context, as described in Section 3.3, e.g., on the location of the cup. We first evaluate our implementation on a simple toy example consisting of only one joint and one context. Thereafter, we evaluate the ability of contextual ProMPs to generalize to new locations of the cup on the real robot.

4.3.1 Toy Example with Two Joints

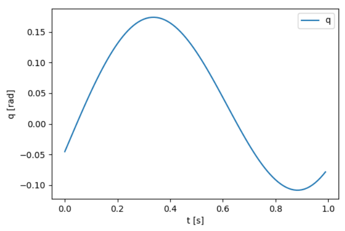
To test our implementation of contextual ProMPs, we evaluated it on a simple toy example with one joint q and one context x, i.e., $\tau = [q, x]^T$. The joint trajectory is a sinus signal deformed with the random noise $\mathcal{N}(\mu, \Sigma)$, i.e., $q = \sin(\theta) + \mathcal{N}(\mu, \Sigma)$.The context is a random constant sampled from the same distribution $\mathcal{N}(\mu, \Sigma)$, i.e., $x \sim \mathcal{N}(\mu, \Sigma)$. An example set of demonstrations is illustrated in Figure 4a. We then create a ProMP with this set of demonstrations. The resulting trajectory distribution is illustrated in Figure 4b. We condition on the positions $x_{new} \in \{0.05, 0.10, 0.15, 2.0\}$ and plot the results in Figures 4c, 4e. As expected, q stays within the original distribution and clearly correlates with x. We notice also that conditioning has to be done on positions within the distribution of demonstrations or at least close enough to them. Otherwise, the conditioned trajectory of q can become multiple times higher than the demonstrations and, thus, will not be reliable anymore and could potentially damage the real robot. An example of such trajectory is illustrated in Figure 4f.



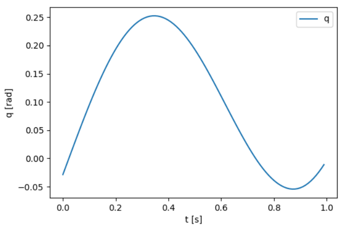
(a) Provided demonstrations, where $q = \sin(\theta) + \mathcal{N}(\mu, \Sigma)$ and $x \sim \mathcal{N}(\mu, \Sigma)$.



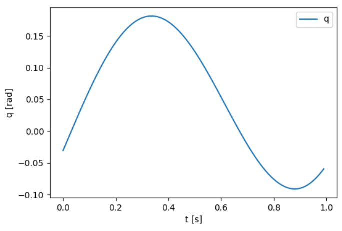
(b) ProMP created from demonstrations illustrating mean and standard deviation.



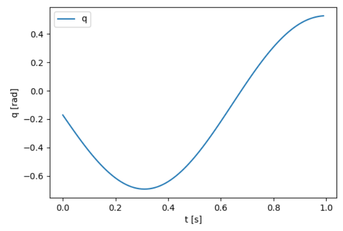
(c) Conditioning on x = 0.10.



(d) Conditioning on x = 0.12.



(e) Conditioning on x = 0.15.



(f) Conditioning on x = 2.0. There is a clear shape difference compared to the previous conditioned trajectories.

Figure 4: A toy example demonstrating the conditioning of a contextual ProMP on a subset.

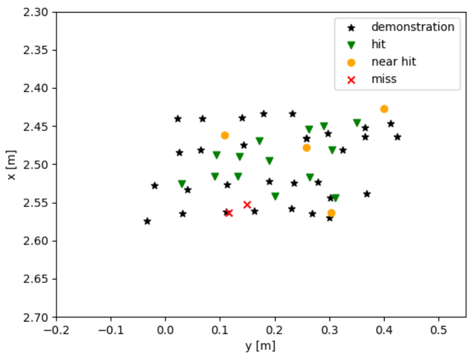
4.3.2 Conditioning Contextual ProMPs on New Locations of the Cup

We obtain a set of demonstrations as described in Section 4.2 and create a contextual ProMP using this set of demonstrations. The cup is then placed on a random position but within the region, which is illustrated in Figure 3b, and condition the contextual ProMP on this position. We repeat these steps

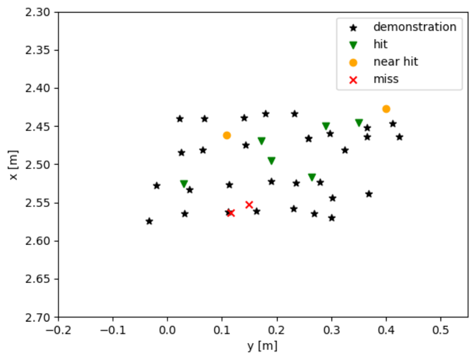
for 20 random positions of the cup, with results illustrated in Figure 5a. The robot could achieve a success rate of 70%. If we count successes along with near misses, the success rate even increases to 90%.
It is clear that contextual ProMPs has difficulties in landing the ball to positions that are almost outside of demonstrations. Both 2 misses, as well as almost all near hits, are at the border of demonstrations. The only near hit inside of the demonstrated region can be explained by a bad quality of the used end effector. We, thus, make a conclusion that contextual ProMPs are very convenient to solve such context problems as beer pong.

4.4 Contextual ProMPs vs. CLR in the Weight-Space of ProMPs vs. CLR in the Weight-Space of DMPs

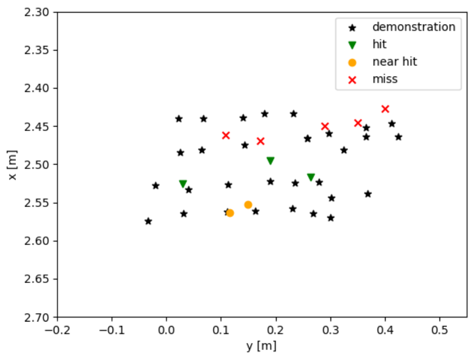
As already mentioned in Section 3.4, there is a relationship between the mean of contextual ProMPs and CLR in the weight-space of ProMPs. As we use the mean of contextual ProMPs to generate trajectories, we want to check this relationship between contextual ProMPs and CLR in the weight-space of ProMPs in experiments. For that, we choose 10 random locations of the cup, record their positions and let both algorithms compute corresponding trajectories. We then let the robot execute the computed trajectories and count successful hits. The results are illustrated in Figure 5.



(a) 20 throwing attempts with contextual ProMP.



(b) 10 throwing attempts with contextual ProMPs.



(c) 10 throwing attempts with CLR to the same locations of the cup as in Figure 5b.

Figure 5: An illustration of throwing attempts with different approaches - contextual ProMPs, CLR in the weight-space of ProMPs and CLR in the weight-space of DMPs. Green markers indicate a successful hit, yellow markers - an almost successful hit, as well as red markers indicate a miss. Grey stars indicate a location for which a demonstration has been provided. Near miss denotes such a shot, that has hit either the edge of the cup or its wall without landing in the cup.

While the contextual ProMP managed to hit the ball in 6 cases and to nearly hit the ball in 2 cases, CLR in the weight-space of ProMPs could only achieve 3 success and 2 near hits. As we can see in Figure 5c, CLR has difficulties in hitting the ball at the borders of demonstration area. Note that in case of contextual ProMP, we approximate locations of the cup using basis functions of ProMPs, while in case of CLR, we use the plain location of the cup. Considering this observation, we actually see that using the plain values of the context does not necessarily lead to a better interpolation. On the contrary, as we have seen in Section 4.3.2, we may produce much better results by approximating the context variables.
In addition, we compare CLR in the weight-space of ProMPs with CLR in the weight-space of DMPs. By that, we compare the ability of CLR to produce the same trajectories in the different weight-spaces. We use the same set of demonstrations to fit the weights of DMPs using Eg. (6). We then condition both algorithms on the same position of the cup and generate the appropriate trajectories. We subtract one trajectory from another, sum the resulting differences along the joint axis for each time step and plot the result in Figure 6.

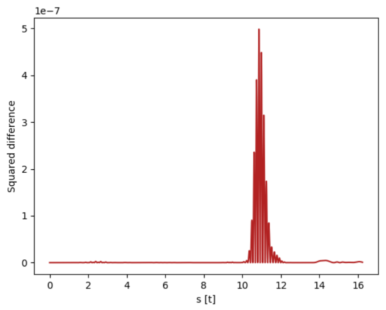


Figure 6: Squared differences between the sum of the joint trajectories produced by CLR in the weight-space of DMPs and ProMPs. Both algorithms have been conditioned on the same location of the cup. A high peak can be observed during the throwing movement, although the difference is actually small.

We observe that although there is a slight difference between trajectories, it is very small to have any impact on the throwing movement. We let the robot execute the first trajectory, place the cup to the position where the ball has landed and let the robot execute the second trajectory. Since the robot has managed to hit the ball exactly to the same location of the cup, we make a conclusion that CLR in the weight-space of ProMPs is similar to CLR in the weight-space of DMPs.

5 Conclusion and Future Work

For the beer pong task, the weights of ProMPs require about 300 radial basis functions to accurately reproduce original demonstrations. A very big number of basis functions means also a very big number of parameters, which makes ProMPs less attractive for policy search methods. However, once reliable and repeatable demonstrations are obtained, contextual ProMPs will already generalize to new cup positions very well. We have proven this statement in our experiments, where the robot could achieve a success rate of about 80%. By that, we suppose that contextual ProMPs can be used in any application task where a context variable is presented, once a reliable hardware is provided.
We have fulfilled our initial target to teach the robot to throw the ball in random locations of the cup. However, there is still some space of improvements. For instance, a good and reliable end effector could have a good impact on the overall performance. We can provide more demonstrations to cover the whole space of the table, so that the robot will have no difficulties in throwing the ball to the borders of the table anymore.
Ideally, the robot should be able to play in every environment as an human does. We, however, have used a model-free imitation learning approach which is not able to generalize to new environments. That is, once the experimental setup is changed, the old demonstrations will not work anymore. We plan to improve this limitation by applying a model-based approach in the same game of beer pong. Finally, we plan to create a beer pong framework with real rules so that a human will be able to play a full game against the robot.

References

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

Stephen Butterworth. On the Theory of Filter Amplifiers. *Wireless Engineer*, 7(6):536–541, 1930.

Andras Kupcsik, Marc Peter Deisenroth, Jan Peters, Ai Poh Loh, Prahlad Vadakkepat, and Gerhard Neumann. Model-based contextual policy search for data-efficient generalization of robot skills. *Artif. Intell.*, 247(C):415–439, June 2017. ISSN 0004-3702. doi: 10.1016/j.artint.2014.11.005. URL `https://doi.org/10.1016/j.artint.2014.11.005`.

G. Maeda, M. Ewerton, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann. Learning interaction for collaborative tasks with probabilistic movement primitives. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 527–534, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041413.

Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic Movement Primitives. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 2616–2624, USA, 2013. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999792.2999904`.

Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, Jul 2017. ISSN 1573-7527. doi: 10.1007/s10514-017-9648-7. URL `https://doi.org/10.1007/s10514-017-9648-7`.

S. Schaal. Dynamic Movement Primitives - A Framework for Motor Control in Humans and Humanoid Robots. In *The International Symposium on Adaptive Motion of Animals and Machines*, Kyoto, Japan, March 4-8, 2003, March 2003. URL `http://www-clmc.usc.edu/publications/S/schaal-AMAM2003.pdf`.

F. Wagner and F. Schmitt. *Robot Beerpong: Model-Based Learning for Shifting Targets*. PhD thesis, TU Darmstadt, 2013. URL `http://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/RobotLearningProject/Wagner_Schmitt_PPPRL_2013.pdf`.

A. Wieland and D. Hoppe. *Comparison of Different Learning Algorithms for Beer-Pong in SL*. PhD thesis, TU Darmstadt, 2013. URL `http://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/RobotLearningProject/WielandEtAl_RLPP_2013.pdf`.